Neural Network Language Models for Candidate Scoring in Hybrid Multi-System Machine Translation

Matīss Rikters
University of Latvia,
19 Raina Blvd.,
Riga, Latvia
matiss@lielakeda.lv

Abstract

This paper presents the comparison of how using different neural network based language modelling tools for selecting the best candidate fragments affects the final output translation quality in a hybrid multi-system machine translation setup. Experiments were conducted by comparing perplexity and BLEU scores on common test cases using the same training data set. A 12-gram statistical language model was selected as a baseline to oppose three neural network based models of different characteristics. The models were integrated in a hybrid system that depends on the perplexity score of a sentence fragment to produce the best fitting translations. The results show a correlation between language model perplexity and BLEU scores as well as overall improvements in BLEU.

1 Introduction

Multi-system machine translation (MT) is a subset of hybrid MT where multiple MT systems are combined in a single system in order to boost the accuracy and fluency of the translations. It is also referred to as multi-engine MT, MT coupling or just MT system combination. Some recent open-source multi-system MT (MSMT) approaches tend to use statistical language models (LMs) for scoring and comparing candidate translations or translation fragments. It is understandable, because the statistical approaches have been dominant for the past decades. Whereas lately, neural networks (NNs) have been showing increasingly greater potential in modelling long distance dependencies in data when compared to state of the art statistical models. Therefore, the aim of this research is to utilise this potential in combining translations.

Since LMs are probability distributions over sequences of words, they are a great tool for estimating the relative likelihood of whether some sequence of words belongs to a certain language. Sentence perplexity – a probability score that can be generated by querying a LM – has been proven to correlate with human judgments close to the BLEU score (Papineni et al., 2002), that has become the main metric for scoring MT, and is a good evaluation method for MT without reference translations (Gamon, et al., 2005). It has been also used in other previous attempts of MSMT to score output from different MT engines as mentioned by Callison-Burch et al. (2001) and Akiba et al. (2002).

Most recently, different order LMs have been used in open-source MSMT approaches like ChunkMT (Rikters and Skadiņa, 2016). This system and the statistical model from KenLM (Heafield, 2011) that it uses will be treated as the baseline for further experiments.

This paper presents an enrichment of the existing MSMT tool with the addition of neural language models. The experiments described use multiple combinations of outputs from online MT sources. Ex-

This work is licenced under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

Proceedings of the Sixth Workshop on Hybrid Approaches to Translation,
pages 8–15, Osaka, Japan, December 11, 2016.

periments described in this paper are performed for English-Latvian. Translating from and to other languages is supported, but it has some limitations as described in the original paper. The code of the developed system is freely available at GitHub[1].

The structure of this paper is as following: Section 2 summarizes related work. Section 3 describes the architecture of the baseline system. Section 4 outlines the LM toolkits that are used in the experiments and section 5 provides the experiment setup and results. Finally, conclusions and aims for further directions of work are summarized.

2 Related Work

Ahsan and Kolachina (2010) describe a way of combining SMT and RBMT systems in multiple setups where each one had input from the SMT system added in a different phase of the RBMT system.

Barrault (2010) describes a MT system combination method where he combines multiple confusion networks of 1-best hypotheses from MT systems into one lattice and uses a language model for decoding the lattice to generate the best hypothesis.

Mellebeek et al. (2006) introduced a hybrid MT system that utilised online MT engines for MSMT. Their system at first attempts to split sentences into smaller parts for easier translation by the means of syntactic analysis, then translate each part with each individual MT system while also providing some context, and finally recompose the output from the best scored translations of each part (they use three heuristics for selecting the best translation).

Freitag et al. (2015) use a combination of a confusion network and a neural network model. A feed-forward neural network is trained to improve upon the traditional binary voting model of the confusion network. This gives the confusion network the option to prefer other systems at different positions even in the same sentence.

3 System Architecture

The main workflow consists of three main constituents – 1) pre-processing of the source sentences, 2) the acquisition of translations and 3) post-processing - selection of the best-translated chunks and creation of MT output. A visualisation of the whole workflow is presented in Figure 1. It outlines the main constituents and sketches their internals.

Going into more detail on the chunking part of the pre-processing step, Figure 2 represents the basic workflow for that. The syntax tree of a sentence is traversed bottom-up, right to left and combines smaller subtrees with bigger ones when possible thereby creating chunks that are no longer than a quarter of tokens or words in the sentence. This specific maximum length for chunks was chosen in previous experiments that showed a general decrease of translation quality or no changes at all for longer maximum chunks. However, if the chunker returns a high amount of chunks for a single sentence, this maximum ratio can be adjusted further. More details on the chunking can be found in the paper of Rikters and Skadiņa (2016) and Rikters (2016).

For translation, several online MT systems are used. The paper of the baseline system described using *Google Translate*[2], *Bing Translator*[3], *Yandex Translate*[4] and *Hugo*[5]. Source languages require compliance with Berkeley Parser (Petrov et al., 2006) parse grammars. The parser is able to learn new grammars from treebanks. Target languages require a language model that is compliant with either KenLM or one of the NN LM tools. New LMs can also be trained using monolingual plain text files as input.

[1] Machine translation system combination using neural network language models - https://github.com/M4t1ss/Batch-ChunkCombiner
[2] Google Translate API - https://cloud.google.com/translate/
[3] Microsoft Translator Text API - https://www.microsoft.com/en-us/translator/translatorapi.aspx
[4] Yandex Translate API - https://tech.yandex.com/translate/
[5] Latvian public administration machine translation service API - http://hugo.lv/TranslationAPI

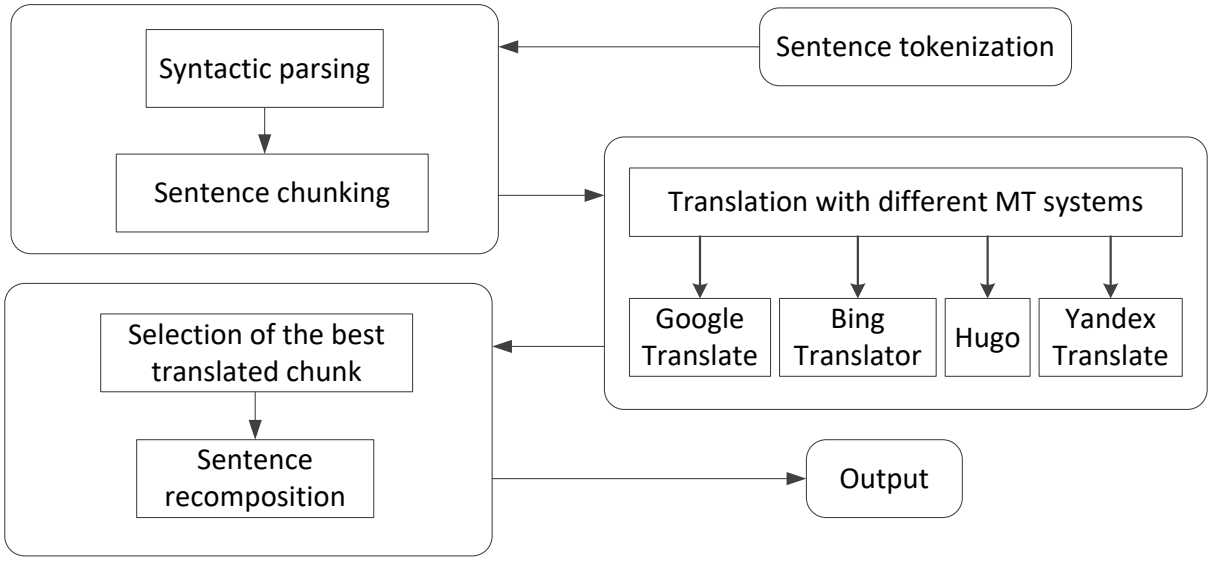

Figure 1. General workflow of the translation process. (Rikters and Skadiņa 2016)

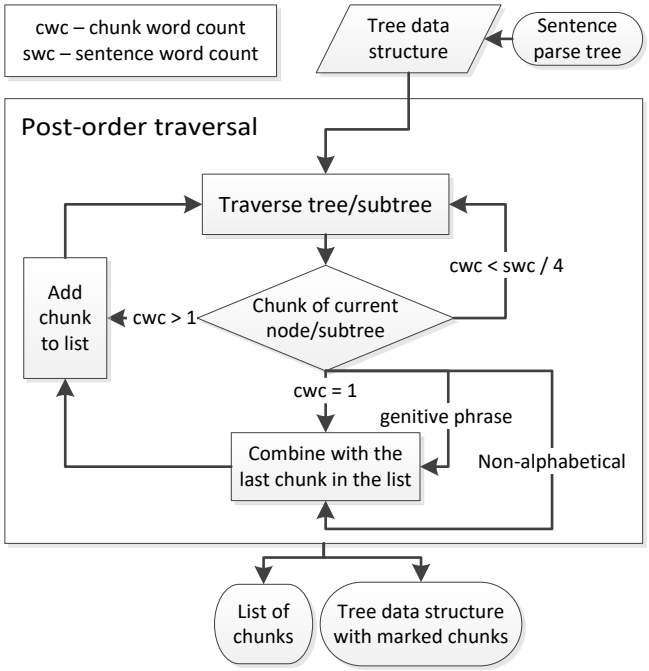

Figure 2. Illustration of how chunks are selected

4 Language Models

4.1 Baseline

The baseline language model was trained with the statistical LM toolkit – KenLM. It is an open-source tool for fast and scalable estimation, filtering, and querying of language models. It is one of the most popular LM tools and is integrated into many phrase-based MT systems like Moses (Koehn et al., 2007), cdec (Dyer et al., 2010), and Joshua (Li et al., 2009). It does the job quite efficiently, thus, it was included as the only LM option in the baseline system. For training, a large order of 12 was chosen for maximum quality.

4.2 RWTHLM

RWTHLM is a toolkit for training many different types of neural network language models (Sundermeyer et al., 2014). It has support for feed-forward, recurrent and long short-term memory NNs. While training different NN configurations, the best results were achieved with a model consisting of one feed-

forward input layer with a 3-word history, followed by one linear layer of 200 neurons with sigmoid activation function.

4.3 MemN2N

MemN2N trains an end-to-end memory network (Sainbayar et al., 2015) model for language modelling. It is a neural network with a recurrent attention model over a possibly large external memory with architecture of a memory network. Because it is trained end-to-end, the approach requires significantly less supervision during training.

MemN2N requires Torch[6] scientific computing framework to be installed for running. Torch is an open source machine learning library that provides a wide range of algorithms for deep learning. For training, the default configuration was used with an internal state dimension of 150, linear part of the state 75 and number of hops set to six.

4.4 Char-RNN

Char-RNN[7] is a multi-layer recurrent neural network for training character-level language models. It has support for recurrent NNs, long short-term memory (LSTM) and rated recurrent units.

To run Char-RNN on a CPU, a minimum installation of Torch is also required. Running on a GPU requires some additional Torch packages. The best scoring model was trained using 2 LSTM layers with 1,024 neurons each and the dropout parameter set to 0.5.

4.5 Environment

The translation experiments were carried out on Ubuntu server with 16GB RAM and 4 cores. This was sufficient because querying the models requires far less computation power than training.

Experiments for LM training and perplexity evaluation were done on three desktop workstation machines with different configurations. The KenLM and RWTHLM models were trained on an 8-core CPU with 16GB of RAM. For training MemN2N a GeForce Titan X (12GB memory, 3,072 CUDA cores) GPU with a 12-core CPU and 64GB RAM. The Char-RNN model was trained on a Radeon HD 7950 (3GB memory, 1,792 cores) GPU with an 8-core CPU and 16GB RAM.

5 Experiments

5.1 Data

To train the LMs the Latvian monolingual part of the DGT-TM (Steinberger et al., 2013) was used. It consists of 3.1 million legal domain sentences. In the case of training an LM with Char-RNN only the first half of this corpus (1.5 million sentences) was used in order to speed up the training process as well as because the character level model requires much less training data when compared with the others. When training all NN LMs evaluation and validation datasets were automatically derived from the training data with the proportion of 97% for training, 1.5% for validation and 1.5% for testing. The final evaluation data consisted of 1,134 sentences randomly selected out of a different legal domain corpus – the JRC Acquis corpus version 3.0 (Steinberger et al., 2006).

The translation experiments were conducted on the English – Latvian part of the JRC Acquis corpus from which both the test data and data for training of the language model were retrieved. The test data contained 1,581 randomly selected legal domain sentences.

For testing on a general domain, the ACCURAT balanced evaluation corpus (Skadiņš et al., 2010) was selected. The general domain test data consists of 512 sentences.

A 12-gram language model for the baseline was trained using KenLM.

[6] A scientific computing framework for luajit - http://torch.ch

[7] Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch https://github.com/karpathy/char-rnn

5.2 Language Modelling Experiments

To justify using different language modelling approaches, different language models were trained with the same and similar (half of the corpus in one case) training data. Table 1 shows differences in perplexity evaluations that outline the superiority of NN LMs. It also shows that the statistical model is much faster to train on a CPU and that NN LMs train more efficiently on GPUs.

System	Perplexity	Training corpus size	Trained on	Training time	BLEU
KenLM	34.67	3.1M	CPU	1 hour	19.23
RWTHLM	136.47	3.1M	CPU	7 days	18.78
MemN2N	25.77	3.1M	GPU	4 days	18.81
Char-RNN	24.46	1.5M	GPU	2 days	19.53

Table 1. Results of language model perplexity experiments.

Since Char-RNN achieved the best results, several in-depth experiments were conducted using just this tool with varying training dataset sizes (for faster training) and NN layer combinations. Figure 3 shows how the network evolves in a setup with two 512-neuron layers. This experiment was conducted on a smaller dataset – only 1/6th of the corpus – allowing it to run for more epochs without early stopping. The perplexity on test data gradually decreased, reaching a lowest score of 22.18.

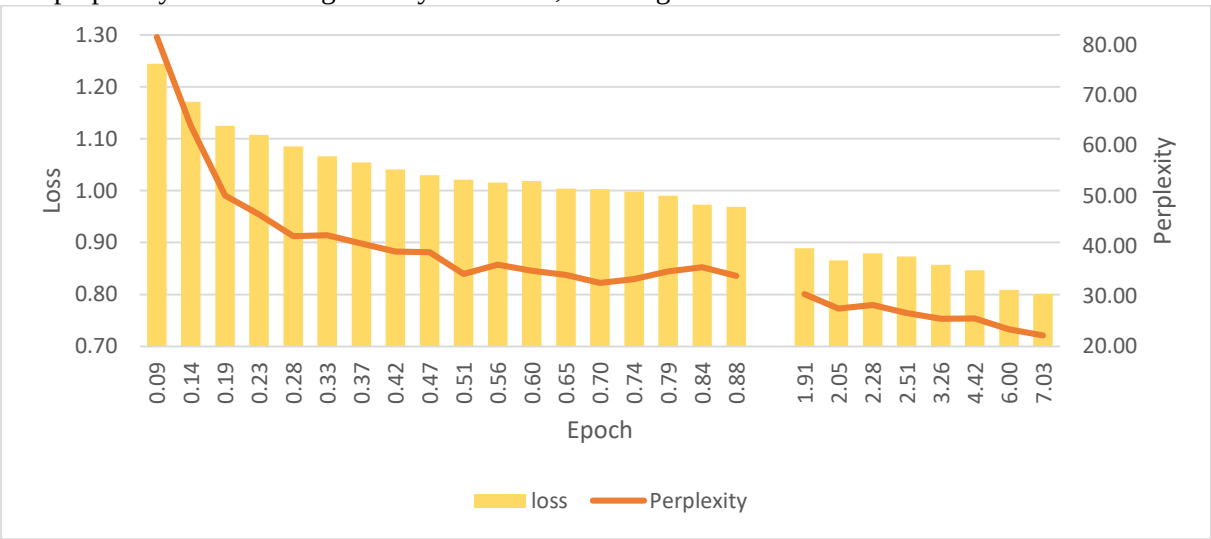

Figure 3. Changes of training loss and perplexity when training a two-layer Char-RNN with 512 neurons on 500 000 sentences.

Another variation for training a LM with Char-RNN is shown in Figure 4. Here 1/3rd of the corpus was used to train a 3-layer RNN with 1,024 neurons per layer. The lowest achieved perplexity was 21.23 after training one day on a GPU.

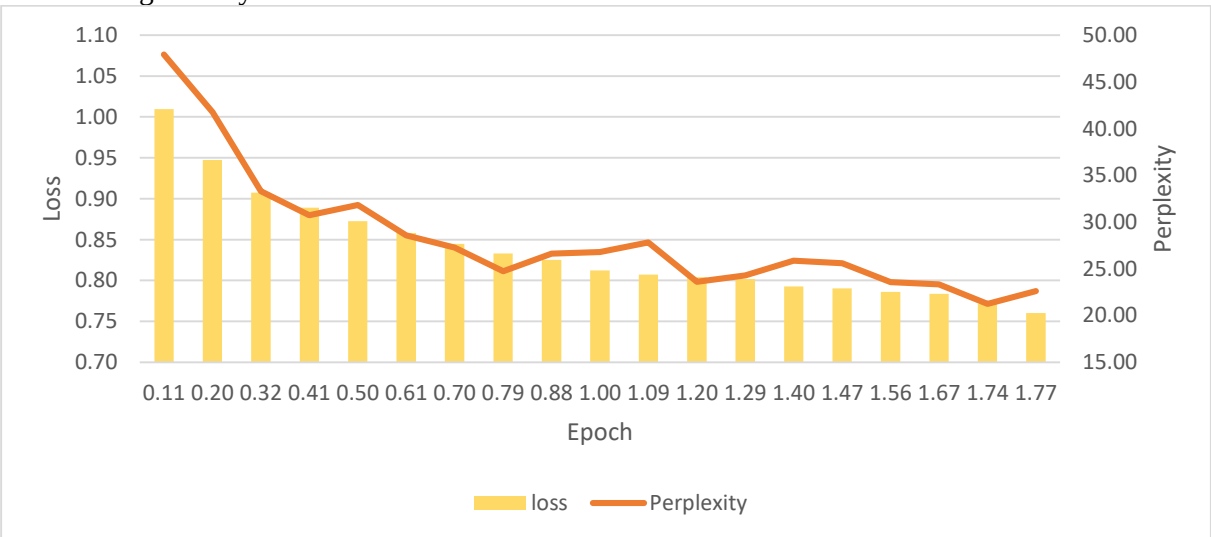

Figure 4. Changes of training loss and perplexity when training a three-layer Char-RNN with 1024 neurons on 1 million sentences

5.3 Machine Translation Experiments

The last column of Table 1 shows differences in BLEU scores when NN LMs were used. Correlation between LM perplexity and the resulting BLEU score is visible as well as a slight improvement in the overall result. Again, due to the outstanding scores of Char-RNN models, they were inspected closer to see how BLEU changes along with perplexity.

The following charts show how perplexity correlates with BLEU in translation test cases on the general domain and legal domain test datasets. Figure 5 represents results from evaluating a combination of Google and Bing (BG) online MT translations (denoted with darker blue colours) and a combination of Hugo and Yandex (HY) online MT (brighter blue colours) on the general domain test dataset. The trend lines (dotted) indicate that for this dataset the combination of BG stays mostly stable but the combination of HY gradually improves as the perplexity of the LM gets lower.

Whereas Figure 6 shows results of combining the same MT on the legal domain test dataset. In this case, while perplexity becomes lower at each time step, the linear trend line for BLEU score of the BG hybrid system does not show a tendency towards climbing higher. As opposed to the BLEU score trend line for HY hybrid system, that showcases improvement along with perplexity.

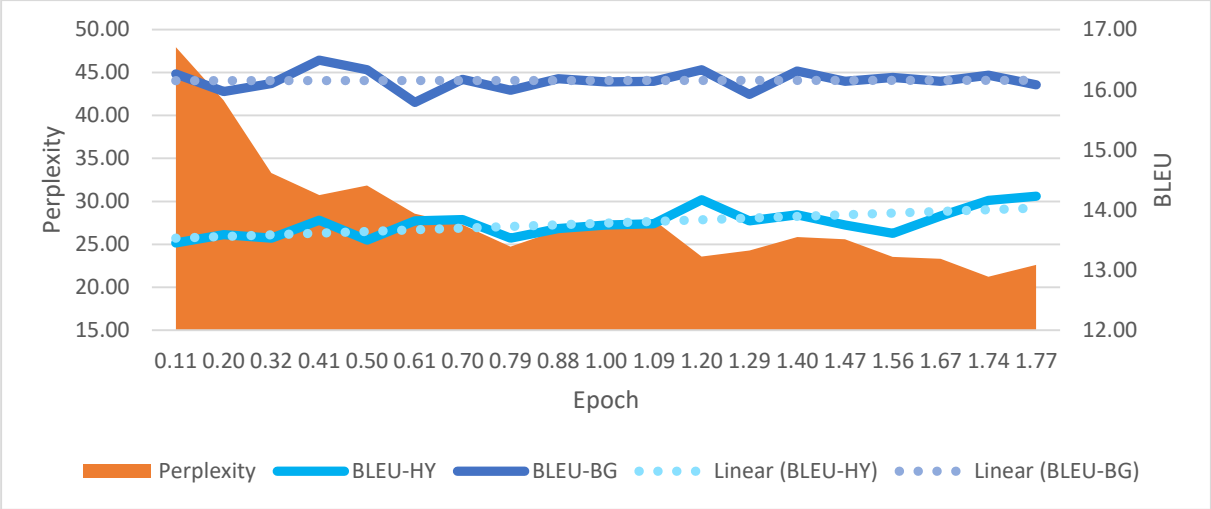

Figure 5. Changes of perplexity when training a three-layer Char-RNN with 1,024 neurons on 1 million sentences and its effect on BLEU score when used in MSMT for combining Bing and Google (BG); Hugo and Yandex (HY) on the general domain test dataset.

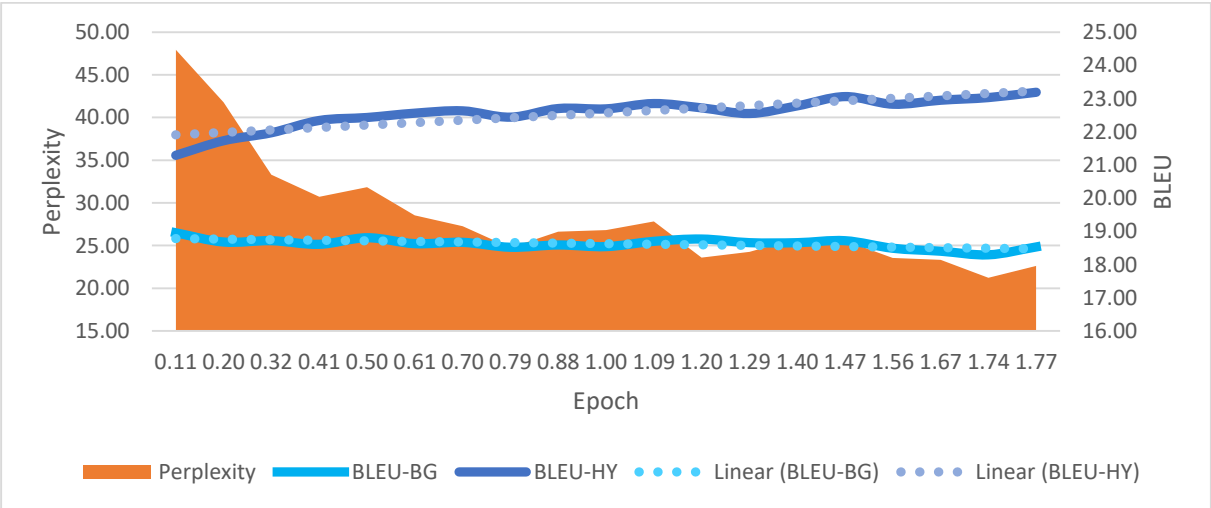

Figure 6. Changes of perplexity when training a three-layer Char-RNN with 1,024 neurons on 1 million sentences and its effect on BLEU score when used in MSMT for combining Bing and Google (BG); Hugo and Yandex (HY) on the legal domain test dataset.

Conclusion

This paper described ways to improve the baseline MSMT system with neural network language models. The main goals were to provide more options for language modelling in the translation combination tool

and to improve translation quality over the baseline. Test cases showed an improvement in BLEU score, when used only with Google and Bing, of 0.35 BLEU points.

In the detailed translation experiments where a BLEU score was obtained in every stage of the LM training there was only a steady correlation of BLEU and perplexity in the case of using Hugo and Yandex translations, which were very different (0.52 – 1.10 BLEU difference with each other) to begin with. In the case of combining Google and Bing translations where the difference was far less significant (0.3 – 0.8 BLEU difference with each other), the BLEU scores of the NN model hybrid were less uniform with perplexity. This indicates that out of very similar options, even the NN model fluctuates with its predictions but it does get more confident in cases where the difference is more obvious.

Adding alternative resources to select from in each step of the translation process could benefit the more advanced user base. For instance, the addition of more online translation APIs like Baidu Translate (Zhongjun, 2015) would expand the variety of choices for translations. A configurable usage of different syntactic parsers like SyntaxNet - Neural Models of Syntax (Andor et al., 2016) is likely to improve the translation process.

Another interesting direction to investigate would be how this system performs when given translations of chunks from locally trained (instead of online) MT systems. For instance, a combination a Moses system with Apertium (Forcada et al., 2011) and even a neural MT system like Nematus (Sennrich et al., 2016).

Reference

Ahsan, A., Kolachina, P.: Coupling Statistical Machine Translation with Rule-based Transfer and Generation, AMTA-The Ninth Conference of the Association for Machine Translation in the Americas. Denver, Colorado (2010)

Akiba, Y., Watanabe, T., Sumita, E.: Using language and translation models to select the best among outputs from multiple MT systems. Proceedings of the 19th international conference on Computational Linguistics-Volume 1. Association for Computational Linguistics. (2002)

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Collins, M.: Globally normalized transition-based neural networks. arXiv preprint arXiv:1603.06042 (2016)

Barrault, L.: MANY: Open source machine translation system combination. The Prague Bulletin of Mathematical Linguistics 93: 147-155. (2010)

Callison-Burch, C., Flournoy, R. S.: A program for automatically selecting the best output from multiple machine translation engines. Proceedings of the Machine Translation Summit VIII. (2001)

Dyer, C., Weese, J., Setiawan, H., Lopez, A., Ture, F., Eidelman, V., Ganitkevitch, J., Blunsom, P., Resnik, P.: cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. Proceedings of the ACL 2010 System Demonstrations. Association for Computational Linguistics, (2010)

Forcada, M.L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J.A., Sánchez-Martínez, F., Ramírez-Sánchez, G. and Tyers, F.M., 2011. Apertium: a free/open-source platform for rule-based machine translation. Machine translation, 25(2), pp.127-144.

Freitag, M., Peter, J., Peitz, S., Feng, M., Ney, H.: Local System Voting Feature for Machine Translation System Combination. In EMNLP 2015 Tenth Workshop on Statistical Machine Translation (WMT 2015), pages 467-–476, Lisbon, Portugal. (2015)

Gamon, M., Aue, A., Smets, M.: Sentence-level MT evaluation without reference translations: Beyond language modeling. Proceedings of EAMT. (2005)

Heafield, K.: KenLM: Faster and smaller language model queries. Proceedings of the Sixth Workshop on Statistical Machine Translation. Association for Computational Linguistics. (2011)

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C.: Moses: Open source toolkit for statistical machine translation. Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, (2007)

Li, Z., Callison-Burch, C., Dyer, C., Ganitkevitch, J., Khudanpur, S., Schwartz, L., Thornton, W.N., Weese, J., Zaidan, O.F.: Joshua: An open source toolkit for parsing-based machine translation. Proceedings of the Fourth Workshop on Statistical Machine Translation. Association for Computational Linguistics, (2009)

Mellebeek, B., Owczarzak, K., Van Genabith, J., Way, A.: Multi-engine machine translation by recursive sentence decomposition. Proceedings of the 7th Conference of the Association for Machine Translation in the Americas, 110-118. (2006)

Papineni, K., Roukos, S., Ward, T., Zhu, W. J.: BLEU: a method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics. (2002)

Petrov, S., Barrett, L., Thibaux, R., Klein, D.: Learning accurate, compact, and interpretable tree annotation. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics. (2006)

Rikters, M.: K-Translate-Interactive Multi-system Machine Translation. International Baltic Conference on Databases and Information Systems. Springer International Publishing (2016)

Rikters, M., Skadiņa, I.: Combining machine translated sentence chunks from multiple MT systems. *CICLing 2016*. (2016)

Sainbayar, S., Weston, J., Fergus, R.: End-to-end memory networks. Advances in neural information processing systems. (2015)

Sennrich, R., Haddow, B. and Birch, A., 2016. Edinburgh Neural Machine Translation Systems for WMT 16. arXiv preprint arXiv:1606.02891. (2016)

Skadiņš, R., Goba, K., Šics, V.: Improving SMT for Baltic Languages with Factored Models. Proceedings of the Fourth International Conference Baltic HLT 2010, Frontiers in Artificial Intelligence and Applications, Vol. 2192., 125-132. (2010)

Steinberger, R., Eisele, A., Klocek, S., Pilos, S., Schlüter, P.: Dgt-tm: A freely available translation memory in 22 languages. arXiv preprint arXiv:1309.5226. (2013)

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., Varga, D.: The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. arXiv preprint cs/0609058. (2006)

Sundermeyer, M., Schlüter, R., Ney, H.: rwthlm-the RWTH aachen university neural network language modeling toolkit. INTERSPEECH. (2014)

Zhongjun, H. E.: Baidu Translate: Research and Products. ACL-IJCNLP 2015 (2015): 61.